UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Division Multidisciplinaria en Ciudad Universitaria

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

DESARROLLO DE UN VIDEOJUEGO COLABORATIVO PARA LA ENSEÑANZA DE LA PROGRAMACIÓN DE COMPUTADORAS

Reporte Técnico de Investigación presentado por:

Saúl Núñez Castruita 150424

Requisito para la obtención del título de:

INGENIERO EN SISTEMAS COMPUTACIONALES/SOFTWARE

Asesor:

M.A.T.I Abraham López Nájera

Dr. Luis Carlos Méndez González

Ciudad Juárez, Chihuahua 3 de agosto de 2020

Ciudad Juárez, Chihuahua, a 3 de Agosto de 2020

Asunto: Liberación de Asesoría

Mtro. Ismael Canales Valdiviezo
Jefe del Departamento de Ingeniería
Eléctrica y Computación
Presente.-

Por medio de la presente me permito comunicarle que, después de haber realizado las asesorías correspondientes al reporte técnico DESARROLLO DE UN VIDEOJUEGO COLABORATIVO PARA LA ENSEÑANZA DE LA PROGRAMACIÓN DE COMPUTADORAS, del alumno Saúl Núñez Castruitade la Licenciatura en Ingeniería en Sistemas Computacionales, considero que lo ha concluido satisfactoriamente, por lo que puede continuar con los trámites de titulación intracurricular.

Sin más por el momento, reciba un cordial saludo.

Atentamente:

Abraham López Nájera
Profesor Investigador

Ccp:
Coordinador del Programa de Sistemas Computacionales
Saúl Núñez Castruita
Archivo

Ciudad Juárez, Chihuahua, a 3 de Agosto de 2020

Asunto: Autorización de publicación

C. Saúl Núñez Castruita

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la publicación del documento de DESARROLLO DE UN VIDEOJUEGO COLABORATIVO PARA LA ENSEÑANZA DE LA PROGRAMACIÓN DE COMPUTADORAS, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.

Dr. Nombre del profesor de la materia

Profesor Titular de Seminario de Titulación II

Declaración de Originalidad

Yo, Saúl Núñez Castruitadeclaro que el material contenido en esta publicación fue elaborado con la revisión de los documentos que se mencionan en el capítulo de Bibliografía, y que la solución obtenida es original y no ha sido copiada de ninguna otra fuente, ni ha sido usada para obtener otro título o reconocimiento en otra institución de educación superior.

Saúl Núñez Castruita

Agradecimientos

[Sustituye este texto escribiendo tus agradecimientos. La sección de agradecimientos reconoce la ayuda de personas e instituciones que aportaron significativamente al desarrollo de la investigación. No te debes exceder en los agradecimientos; agradece sólo las contribuciones realmente importantes, las menos importantes pueden agradecerse personalmente. El nombre de la agencia que financió la investigación y el número de la subvención deben incluirse en esta sección. Generalmente no se agradecen las contribuciones que son parte de una labor rutinaria o que se reciben a cambio de pago.

Las contribuciones siguientes ameritan un agradecimiento pero no justifican la coautoría del artículo: ayuda técnica de laboratorio, préstamo de literatura y equipo, compañía y ayuda durante viajes al campo, asistencia con la preparación de tablas e ilustraciones o figuras, sugerencias para el desarrollo de la investigación, ideas para explicar los resultados, revisión del manuscrito y apoyo económico".]

Dedicatoria

[Aquí escribe tu dedicatoria.]

Índice general

Índice de figuras

Índice de tablas

Resumen

[Sustituye este texto escribiendo tu sinopsis o resumen. Es un panorama general de todo lo que el lector encontrará en tu documento, en no más de una página. Recuerda que este, junto con el título, son la parte más leída de tu documento cuando alguien más lo busca en las bases de datos, el "punto de venta".]

Introducción

[Redacte de manera coherente en una cuartilla cuál es la nueva contribución, su importancia y por qué es adecuado para sistemas computacionales. Se sugiere para su redacción seguir los cinco pasos siguientes: 1) Establezca el campo de investigación al que pertenece el proyecto, 2) describa los aspectos del problema que ya han sido estudiado por otros investigadores, 3) explique el área de oportunidad que pretende cubrir el proyecto propuesto, 4) describa el producto obtenido y 5) proporcione el valor positivo de proyecto.]

Capítulo 1
Planteamiento del Problema

En este capítulo se entrara en detalle sobre los términos relacionados al proyecto y analizamos trabajos anteriores sobre software enfocados en la enseñanza de programación de computadoras desde los 60s hasta la actualidad. De igual manera, se define que atacaremos con este proyecto, una problemática y que objetivos tiene el proyecto.

1.1. Antecedentes

Videojuegos como método de enseñanza

El área de serious games puede pensarse como dos grandes áreas: entre ellas existe el edutainment, aplicaciones informáticas con animaciones, elementos multimedia que muestran la información de una manera divertida; y los videojuegos, creados con la pretensión explicita de enseñar, incluyen el entrenamiento a base de simuladores, la transmisión de la información o incluso, la promoción de alguna idea o marca [**?**]. La principal diferencia para es la difusión del contenido, el edutainment prioriza la difusión del contenido de manera lúdica, mientras que los videojuegos para enseñar deciden sacrificar la parte lúdica para explorar de mejor manera los conocimientos y lo hace de maneras más complejas [**?**]. Mucho desarrollo en el área y uno de los primeros en adoptar los serious games fue la milicia, se usan para el adies-

tramiento militar, sin contar con el número de simulaciones de distintos vehículos militares [?]. Asimismo, han existido serious games de política, con diferentes propósitos, como la comunicación de ideas, criticar a oponentes o reproducir sus discursos, algunos ejemplos son los news games que mezclan lo periodístico con la denuncia política y exponen una problemática de un determinado lugar y tiempo. Adicionalmente existen géneros como los advergames, usados para publicidad, los de salud que permiten a los estudiantes aprender sin temor a equivocarse con un ser humano, los videojuegos artísticos y los videojuegos religiosos [?].

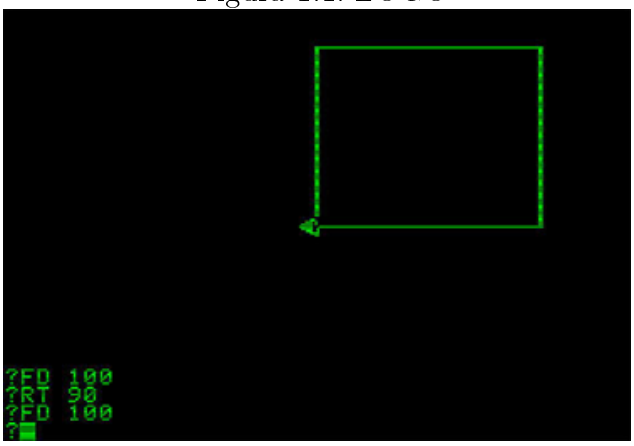
Evolución de la enseñanza de la programación

Podemos notar como punto importante la creación de Pascal a finales de los 60s y principios de los 70s por parte de Niklaus Wirth, un lenguaje enfocado en la enseñanza de técnicas de programación, con una nueva metodología de programación denominada programación estructurada [2]. Sin embargo, no todo es tan claro, no ha existido un consenso de los métodos a utilizar y no ha existido alguno que se haya impuesto. Existen métodos para la enseñanza de otros paradigmas de programación como el funcional, el imperativo o el imperativo orientado a objetos; y dentro de estas divisiones hay quienes enseñan a programar mediante un lenguaje particular o quienes emplean un lenguaje algorítmico que pudiera adaptarse a otros lenguajes de programación [2]. Sin embargo, podemos destacar la existencia de algunas nuevas herramientas:

LOGO

Logo es un lenguaje de programación diseñado como una herramienta para aprender programación. Permite mover una tortuga (originalmente una creatura robótica) a cualquier dirección mediante introducir comandos a la computadora [?] cómo se nota en la 1.1.

Figura 1.1: LOGO



Pascal

Lenguaje creado en 1970 por Niklaus Wirth para enseñar programación estructurada, paradigma de programación donde se pone énfasis en las estructuras condicionales y cíclicas, sin GOTO [?]. Fue comúnmente usado durante después de la primera mitad de los 70s y los 80s debido a que estaba disponible en un gran número de computadoras, así como por su claridad y seguridad, no era raro que fuera usado para la producción de software.

Python

Diseñado por Guido Van Rossum originalmente estaba pensado para como un lenguaje de scripting para el sistema operativo Amoeba [?]. Su ventaja en el ámbito educativo es su facilidad de uso, comúnmente descrito como poder programar en inglés y la gran cantidad de librerías que permiten crear cosas complejas de manera rápida.

Lego WeDo 2.0

Es un kit diseñado para introducir a niños y niñas a la robótica, construyen pequeños robots con LEGO y de diferentes formas pueden programarlos para que se muevan. Normal-

Figura 1.2: El entorno de programación Scratch

mente hacen uso de programación por bloques.

1.1.1. Trabajos relacionados

En la actualidad, muchos programas para enseñar programación utilizan drag-and-drop para realizar la tarea de programación, a esta se le conoce como block-based programming [?]. Ejemplos de herramientas de esta categoría incluyen el entorno Scratch, que permite crear pequeños juegos o animaciones con sus herramientas, como se nota en la 1.2 y Code.org, donde hay diversas actividades para aprender a programar.

Similar, aunque a diferencia de este proyecto, Lego WeDo va orientado a la robótica. Según la clase, eligen de una de las diferentes opciones disponibles para construir y los alumnos siguen unas instrucciones similares a otros kits de LEGO, y puede ser programado en Scratch o en un software propio de LEGO [?]. Incluye un hub con conectividad bluetooth, donde se pueden conectar sensores y motores [?], como se puede ver en la 1.3. El software de LEGO es igual, block-based programming e incluye sesiones donde los estudiantes aprenden de diversos temas y realizan actividades de programación relacionadas a estos. A diferencia de estos dos, en lugar de ser el profesor la principal fuerza para poner nuevos retos y enseñar nuevos conceptos, acudiremos a mecánicas vistas en videojuegos.

Figura 1.3: El entorno de programación Scratch

Figura 1.4: Kahoot!

Por otro lado esta "Kahoot"!, que es una plataforma de game-based learning [?], esta permite a profesores poner quizzes, de los cuales los estudiantes eligen la respuesta en sus teléfonos y en una pantalla más grande a la que tiene acceso el profesor se muestra la pregunta actual con sus posibles respuestas, así como la posición en la que están los alumnos entre cada pregunta, como se nota en la 1.4. Este proyecto es útil para las evaluaciones, sin embargo, de manera similar a este el profesor tiene el control del tema a trabajar y lo controla desde un lobby.

Hay una variedad de juegos de programación, como CodingGames, que es un sitio de minijuegos donde se ofrecen instrucciones sobre el punto del juego y cuenta con un IDE para programar una solución para pasarlo. Asimismo, CodeCombat, un juego de dungeon crawling donde se debe programar para avanzar en el juego [10], hay aspecto multijugador y al igual a "Kahoot!" uno puede ser parte de un lobby de una clase y el profesor puede ver el avance de sus alumnos, pero cada jugador avanza a su ritmo. Similarmente, en Screeps un MMO de estrategia para programadores de Javascript y CheckIO, se deben resolver problemas en Python [?]. Sin embargo, todos están enfocados además en la enseñanza de un lenguaje de

programación (Python o Javascript son opciones comunes), por lo que para un principiante puede ser un punto de contención adicional el aprender la sintaxis durante la práctica. Similarmente, en el área de edutainment existen juegos como Human Resource Machine, Hack 'n' Slash, TIS-100, donde a excepción del ultimo están enfocados a desarrollar una lógica para programar, TIS-100 agrega un lenguaje ensamblador con programación a base de "nodos", cada nodo solo acepta programas pequeños, se comunican varios nodos para crear programas más complejos, TIS-100 es más apto para enseñar sistemas distribuidos en lugar de programación básica.

1.2. Definición del problema

La enseñanza de programación es regularmente una dualidad con la enseñanza de un lenguaje de programación, donde se enfoca en la enseñanza de la sintaxis y en la enseñanza y práctica de hilar la lógica necesaria para crear un programa para computadoras, donde entran técnicas como [12]:

- Descomposición de problemas complejos
- Reconocimiento de patrones para buscar similitudes entre problemas
- Abstracciones
- Algoritmos

La enseñanza de programación es un proceso complejo, donde el alumno tiene que acostumbrarse a razonar la forma de solucionar un problema (a base de descomposición) y aprender un lenguaje de programación con todas las idiosincrasias que este tenga, desarrollando los debidos modelos mentales sobre el funcionamiento del lenguaje y sobre la notación para resolver un problema [**?**]. La sintaxis de un lenguaje de programación puede ser complicado

para un principiante, se ha encontrado que estos tienen dificultad para encontrar errores gramaticales en su código. Para esto, sería mejor dotar a alguien aprendiendo a programar con la habilidad de razonar problemas en un entorno donde no tengan que razonar mucho la sintaxis del lenguaje, y haya aspectos visuales que permitan al usuario experimentar programando y desarrollar los modelos mentales, pero reduzcan la carga cognitiva de usar una herramienta como un lenguaje de programación escrito por texto.

1.3. objetivo general

Desarrollar un videojuego multijugador colaborativo que permita a alumnos aprender programación estructurada usando un lenguaje de programación visual.

1.3.1. Objetivos específicos

1. Creación de un documento de diseño.
    a) Diseño de puzzles en un entorno colaborativo relacionados a distintos conceptos de la programación estructurada.
    b) Crear un mundo virtual que conecte los diversos puzles para enseñar a los alumnos programación
2. Crear un entorno de programación por bloques para que los alumnos puedan programar soluciones de puzzles.
3. Desarrollar un componente de servidor para la creación de lobbies.
4. Desarrollar el software en Javascript para el navegador web.
5. Validar resultados a través de pruebas estadísticas.

1.4. Pregunta de Investigación

¿Qué ventajas existen en la enseñanza de programación mediante el videojuego propuesto?

1.5. justificación

Un videojuego colaborativo permitiría tener ventajas como el desarrollo de las habilidades cognitivas, sociales en un ambiente de colaboración permite a sus integrantes encontrar nuevas formas de atacar problemas, algunas de estas de especial importancia para la programación. Adicionalmente, los videojuegos son peculiarmente efectivos para la resolución de problemas, debido a que permiten "aprender haciendo" donde el proceso de aprendizaje es una constante practica e interacción con tareas más complejas donde los jugadores encuentran reglas subyacentes que son de utilidad para resolver los problemas que nos plantee el juego [**?**]. Peculiarmente, este método de aprendizaje iterativo es muy cercana a la manera en la que resolvemos problemas en la vida diaria, en esta regularmente resolvemos problemas por analogía, donde recurrimos a esquemas (paquetes de información sobre las propiedades del problema como: el conocimiento antes de intentar resolver el problema, a que se quiere llegar, los posibles pasos y aquello que es permitido y las consecuencias de escoger incorrectamente) y se recupera el marco que se creó anteriormente para resolver este nuevo problema. Un videojuego para la enseñanza de programación con los puntos anteriores, puede ser una excelente herramienta para la enseñanza en salones de clase de los elementos básicos de la programación y permitir a principiantes desarrollar la habilidad de resolver problemas. Adicionalmente, un videojuego corto puede ser una excelente herramienta para una clase de Hour of Code (una propuesta para en una hora, introducir a pequeños o a grandes a la computación o a la programación).

1.5.1. Alcances y limitaciones

Alcances

- Este videojuego se podrá jugar desde un navegador
    - Ante ello será programado en el engine Unity3D exportando al entorno WebGL
- El producto consta de un tiempo de juego de una hora a hora y media para ser completado por el jugador.
- Los puzles están enfocados a crear o modificar código con el fin de alterar algún aspecto del nivel, ya sea enemigos, el ambiente o manejar algún NPC (personaje no jugable) para resolverlos.

Limitaciones

- Se tuvo un límite de tiempo de desarrollo a 4 meses para acomodar la validación y procesos adicionales.
- Se uso la librería Blocky para implementar la programación por bloques, está pendiente descubrir debilidades en sus capacidades, por lo que puede cambiar temas jugables, los tipos de puzzles o jugabilidad general según se encuentren limitaciones.
- Se uso un servidor ASP.NET Core para el desarrollo del modo multijugador.

Capítulo 2
Marco teórico

En esta sección entramos en detalle el marco conceptual y el marco tecnológico. En la primera sección se verá la forma de aprovechar los videojuegos para la enseñanza, sus ventajas y la forma de enseñanza que comúnmente aprovechamos con estos, además tocamos temas relacionados a la programación por bloques y la metodología de software a usar y sus artefactos. En la segunda sección discutimos de tecnologías aptas para la realización del proyecto.

2.0.1. Marco Conceptual

Game-based learning y educación colaborativa

Game-based learning es "en formación en la cual los contenidos teóricos son presentados por medio de un videojuego" [?] y contienen elementos como [?]:

- Historia para darle inmersión a los jugadores
- Gamificación, como rankings o un sistema de puntos
- Feedback inmediato, algunos juegos tienen información como en que se equivocaron y una oportunidad de realizarlo otra vez

- Simulación de una situación de la vida real, permitiendo la práctica segura en ambientes cercanos a donde aplicarán su conocimiento

Los videojuegos pueden ser una muy poderosa herramienta para el aprendizaje: permiten el aprendizaje *Just In Time* que bajo desafíos realizables empujan al jugador a ser competente y nos ayudan a fomentar el pensamiento crítico [**?**]. El aprendizaje *Just In Time* ofrece el conocimiento necesario para hacer una tarea justo cuando es necesario [**?**]. Hay métodos en los que la enseñanza *JIT* ocurre naturalmente como ver un video de *Youtube* cuando no sabemos cómo realizar una tarea, en los videojuegos es natural cuando hay introducciones a acciones en el juego como la forma en la que lo invocamos o las acciones que realizan, ya sea mostrando una interfaz gráfica con la forma de invocarlo. Se encontró que los estudiantes en el aprendizaje colaborativo los alumnos se enseñan uno al otro al responder a dudas y clarificar preconcepciones, desarrollan comunicación oral y capacidad de liderazgo, aumenta la retención del material, la responsabilidad y expone a los alumnos a perspectivas diversas [**?**]. Se ha encontrado que los videojuegos colaborativos agregan ventajas como el trabajo en equipo, el pensamiento creativo, la comunicación y la colaboración [**?**].

Block-based programming

Block-based programming se ha establecido como la forma de introducir a aquellos que quieren aprender a programar seguido por el éxito de Scratch y Code.org [**?**]. Este método de programación usa piezas de un rompecabezas como metáfora para proveer al usuario pistas sobre cuales instrucciones son válidas [**?**].

Kanban

Kanban es una metodología ágil para el desarrollo de software con un énfasis en la entrega continúa teniendo en cuenta la capacidad del equipo [**?**]. Las métricas de Kanban son las

siguientes [?]:

Figura 2.1: Diagrama de flujo acumulativo

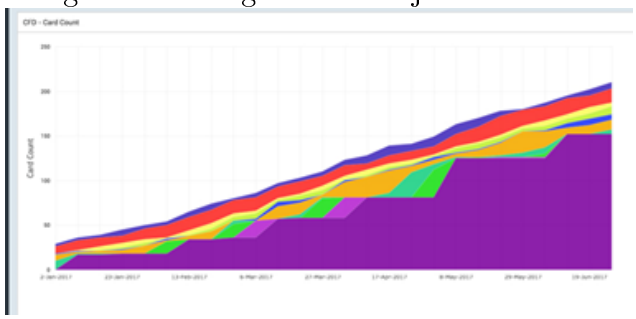
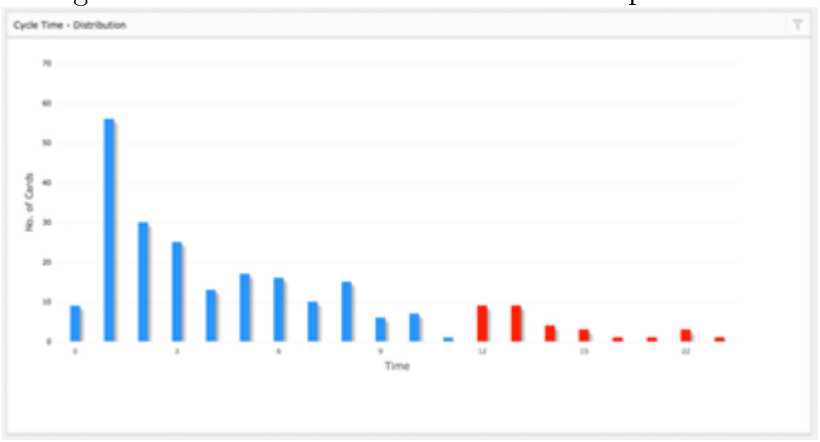


Figura 2.2: Gráfica de distribución de tiempos de ciclo



- **Diagrama de flujo acumulativo 2.1:** Provee información relacionada con la capacidad del equipo. Está basado en tiempo y muestra cómo se mueven las tarjetas de izquierda a derecha en el tablero. La altura de las bandas muestra el número de tarjetas en esa etapa durante cierta unidad de tiempo.

- **Gráfica de distribución de tiempos de ciclo 2.2:** Es útil para ver la frecuencia con la que las tarjetas son completadas a lo largo del tiempo.

2.0.2. Marco tecnológico

Para el desarrollo de este proyecto es necesario tener herramientas para las partidas con otros usuarios, así como manera de usar librerías probadas y comúnmente usadas.

Unity3D

Es un motor que permite diseñar videojuegos. Permite usar C# para programar el juego, compilando, usando Mono. [?] Es un motor muy capaz que permite desarrollar videojuegos 2D y 3D, con una variedad de herramientas para facilitar el desarrollo. Unity puede crear el juego para varios entornos, como consolas de videojuegos como la Nintento Switch, el Xbox One, PS4, Windows, MacOS, Linux, Android, iOS, o la web mediante WebGL [24]. La ventaja del entorno web es la posibilidad de conectar Unity con la pagina web que lo alberga, con esto puede puedo interactuar con elementos HTML o con librerías JS, en este caso lo conectaremos con Blocky.

Blocky

Librería de Google que permite al usuario escribir código usando bloques como se nota en la 2.4 y compila el resultado a diferentes lenguajes entre ellos Javascript, Dart, Python, Lua y PHP.

ASP .NET Core

Framework para sitios web de Microsoft. Será usado como la base para hospedar la API del servidor, servir los archivos del juego, así como páginas de landing y de creación de lobbies.

Figura 2.3: Algunas capacidades de Unity para el desarrollo de videojuegos: animaciones, máquina de estados y para videojuegos 2D, un editor de sprites

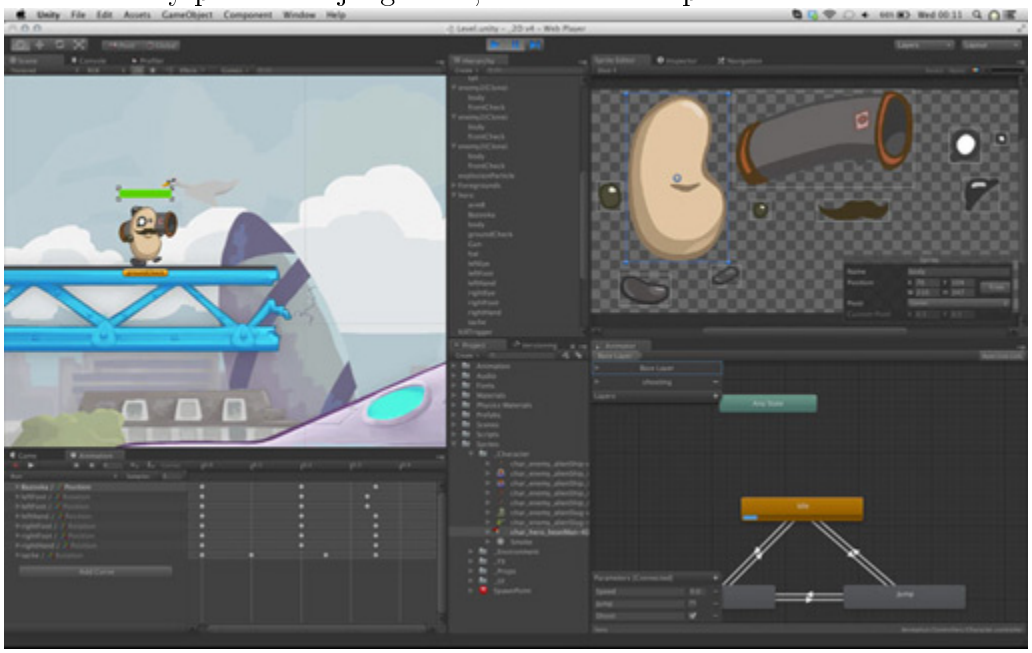
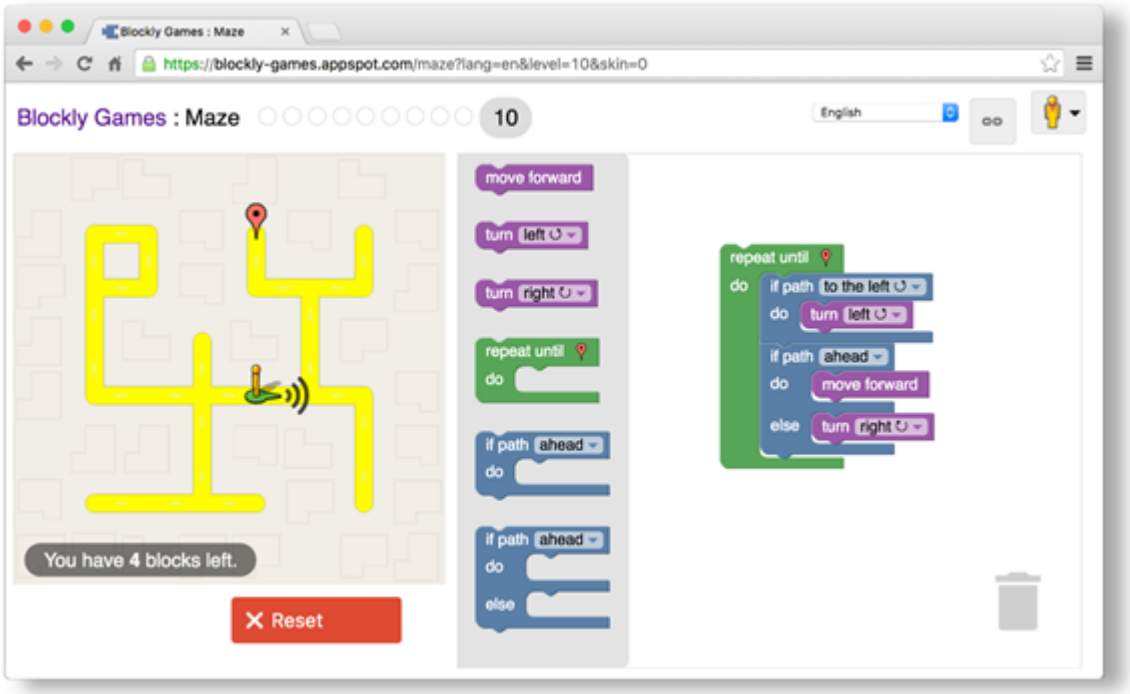


Figura 2.4: Un ejemplo de un juego usando la librería de Blocky



SignalR

Es una librería de Microsoft que nos permite hacer comunicación en tiempo real web. Permite ser usado para crear RPC, o llamadas de procedimiento remoto. Su principal ventaja es que mandar mensajes al servidor es muy fácil a otras soluciones, hay una API definida que la librería se encarga que de que en cada request sea cumplida el schema de la API y se encarga de la serialización y deserialización de cada comunicación. Así como pasar información de cambios al cliente ocurre de manera rápida, maneja el protocolo de transferencia y evita mucha configuración que de otra forma seria necesario.

Git

Git es el sistema de control de versiones más popular del mundo, fue creado en 2005 por Linus Torvalds [27]. Es un ejemplo de un DVCS, un sistema distribuido de control de versiones, a diferencia de sistemas donde en un solo lugar esta todo el historial de versiones como CVS o Subversion, en Git todas las copias funcionales de código son un repositorio que contiene todo el historial de versiones. En comparación con otros sistemas, Git está diseñado para que la creación de ramas y tags sean operaciones baratas, por lo tanto, rápidas.

Github es un servicio de hospedaje de repositorios Git [**?**]. Se uso como *repositorio remoto* para sincronizar cambios, además de aprovechar sus funciones para crear pipelines para el continous deployment.

Capítulo 3
Desarrollo del Proyecto

En este capítulo se discute sobre el trabajo a realizado para la completitud del proyecto, así como la descripción y el post-mortem de las diferentes actividades hechas durante su realización.

3.1. Producto propuesto

El producto desarrollado es un videojuego colaborativo donde las mecánicas estén afines a que los alumnos conozcan diferentes conceptos básicos de los lenguajes de programación y que tengan que hacer y modificar pequeños programas para resolverlas. Este videojuego se enfoca en la enseñanza de las características más elementales de la programación estructural como: variables, condicionales, ciclos y funciones. Los puzzles para la enseñanza de estos conceptos sean pequeños problemas en el mundo de juego donde tendrán que modificar código existente para cambiar su comportamiento o crear todo un programa pequeño que tenga el comportamiento deseado, usando las técnicas que han descubierto, algunos de estos puzzles contienen elementos que obliguen a varios jugadores a colaborar para resolverlos. Estos puzzles tratarán temas de programación estructurada, estás seguirán el material del curso de Fundamentos de la Programación y Programación I, así como los libros Fundamentos de programación: Algoritmos, estructura de datos y objetos de Luis Joyanes Aguilar y Prelude

to Programming: Concepts and design de Steward Venit y Elizabeth Drake. El videojuego solo tendrá un nivel con una duración aproximada de una hora a hora y media.

3.2. Forma de validación

Para la evaluación de la eficacia de este videojuego en la enseñanza de programación se realizarán dos clases con alumnos del campus de Ciudad Universitaria de la UACJ que no hayan visto programación o estén cursando clases como Fundamentos de Programación o Programación I. Estas dos clases tendrán alumnos totalmente distintos. Estas se realizarán de la siguiente manera:

- En la primera clase se colocaron a todos los alumnos en un lobby del videojuego creado y tendrán que completar todo el primer nivel del videojuego con todos los puzzles, el profesor (uno de los integrantes del equipo) resolverá dudas.
- La segunda y similar en tamaño, fue una clase de dos horas aproximadamente donde se enseñarán conceptos de programación estructurada con Scratch, un lenguaje de programación por bloques muy similar a Blocky de Google, se separará en secciones donde cada una alberga un concepto nuevo y una pequeña actividad de 5 a 10 minutos, se tendrá el mismo temario que en el videojuego.

Al terminar se les hizo un pequeño examen donde se verá el conocimiento sobre diferentes cosas que acaban de ver en la última hora y se comparará su rendimiento, estos dos grupos son muestras distintas de una población de todos aquellos universitarios que no saben programar y estas dos clases pueden tener un numero distinto de participantes, además de que se desconoce la forma en la que estarán distribuidos sus resultados, por lo que se usara la prueba de la U de Mann-Whitney, una prueba no paramétrica que permitirá hacer una afirmación sobre una hipótesis nula, en este caso si se ha encontrado mejora en el rendimiento (denotado por su

resultado en el quiz) entre los dos grupos. Adicionalmente, el examen contará con preguntas adicionales sobre la opinión de los estudiantes sobre el método de enseñanza, tales como: si les entretuvo, si se les hizo interesante, cuanto se les hizo que aprendieron en una escala del 1 al 10, y si les intereso la programación y se realizará un análisis comparativo.

3.3. Fases (Metodología)

El desarrollo de este proyecto incluye varias actividades. Incluye planificación del proyecto con la creación de un documento de diseño, creación de *assets*, así como el desarrollo del juego integrando lo creado en los dos puntos mencionados anteriormente. En Kanban estas actividades estarán en tarjetas que serán apiladas según su dependencia causal.

Integralmente, estas se verán de la misma forma que en la Ilustración 9, primero será necesario definir en el documento de diseño que comprende cada puzle, el entorno del juego y lo que se espera que sean los controles del juego; finalmente se podrán crear los distintos puzzles según las características del engine, como sprites, así como la forma de interconectar el código creado para el juego al engine.

3.4. Avances

3.4.1. Documento de diseño del juego

Se trabajó en un documento que detalle los diferentes sistemas del videojuego, así como los diferentes puzzles, el arte del juego, etc. Un documento de diseño incluye información detallada que ayude a definir las labores a realizar durante el desarrollo y dar una explicación inequívoca de la entrega. El diseño del juego se planeó un videojuego estilo juego de mesa, donde los movimientos por el tablero ocurren por el lanzamiento de dados, al aterrizar en un

cuarto la primera vez en el juego hay cuartos dónde como jugador se puede obtener tanto un ítem que ofrece ventajas en el juego o un puzzle que ofrece al jugador una recompensa. Los puzzles son pequeños programas que el jugador tiene que hacer o editar. Estos incluyen temáticas como variables, condicionales, ciclos y funciones. En la x podemos ver la organización de los puzzles, además de la división por temáticas, va escalonada la dificultad, esta dificultad puede ser mayor entre pisos de la mansión del juego o si el jugador ha resuelto un puzle de ese tipo antes en el juego. Los ítems del juego pueden dar al jugador alguno de las siguientes ventajas:

- Cambiar stats propios o de enemigos
- Hacer algo que el jugador normalmente no podría hacer, como moverse entre pisos o entrar a lugares o abrir tesoros
- Cambiar tiros del dado

En algunos cuartos del juego se diseñó para que, como jugador se puede entrar en mini peleas donde hay enemigos que enfrentas con código o preguntas teóricas y al robarle los suficientes puntos al enemigo ganas obteniendo recompensas para usarlas en el juego. En el juego a lo largo de los turnos, hay probabilidad de entrar en el modo de enfrentamiento del final del juego, en este se enfrenta el jugador más fuerte va los demás.

3.4.2. Servidor

Se creo un servidor con*ASP.NET Core* para encargarse de la API *SignalR* que tendra acceso los clientes del juego y las páginas del profesor. Se vio la factibilidad para usar *Node* para ejecutar los programas que hagan los jugadores en el servidor, donde aseguraremos que cumplan el puzzle, se encontro una librería para obtener un entorno protegido para correr código de los jugadores, en ejecución se manda al microservicio el código y detalles sobre

lo que se verificara mediante una *API REST*. Además, tendrá un sistema de lobbies. Para desarrollar este sistema se usará la librería Socket.io en el servidor y en el navegador, donde se conecta y se pasaran mensajes a Unity que correrá el juego en el cliente esperará la respuesta de cualquier acción desde el servidor.

El servidor será autoritativo. Mucho del juego será computado en el servidor, donde el cliente solo estará como un cliente ligero de cierta forma. El cliente se conectará mediante *SignalR*. El diagrama de secuencia UML del servidor interactuando con el cliente se puede ver en

3.4.3. Creación del cliente de juego

Según detalles del documento de diseño, se crearon diversos sprites para elementos del mapa, personajes, animaciones e interfaces gráficas. Esta etapa será encargada de la programación de diversos sistemas y comportamientos de distintos elementos del juego que el usuario podrá interactuar directamente y que correrán en el navegador. Incluye la interconexión con Javascript en el navegador con Unity para el sistema de programación a bloques usando Blocky, este será tratado como una ventana arriba de todo el contenido que puede ser mostrada u ocultada según el estado del juego. Uno de los requisitos es poder pasar archivos XML para definir el entorno del editor de bloques al jugador. El juego estará embebido en una página web con un área central donde vivirá el juego, del cual Unity pide que se le pase un elemento div. Esta API se conectará por medio de un sistema de interoperabilidad del motor Unity que permitirá usarlo en C#.

El juego, entre resolver la información de los cambios de otros clientes y alternar a las entradas del jugador cuando es turno de este, el comportamiento del cliente ante diversos estados se puede ver como una máquina de estados como en la Ilustración 10. Inicia la conexión con el servidor, cuando el jugador introduce un código de lobby existente en la lista de sesiones en el servidor inicia el juego y el resto de los estados se alternan según lo

mencionado anteriormente.

3.4.4. Codificación de *puzzles*

El trabajo de esta sección puede ser divido en dos secciones:

- Microservice para evaluar puzzles
- Tooling para configurar los puzzles

En primera instancia cada puzzle se manda al servidor para ver si fueron respondidos de manera correcta. En este se configurará si hay declaraciones de variables que debe tener el código, así como si la salida del programa y lo que se revisara en el parsing para ver que sigan la estructura correcta. Sin embargo, en el backend del servidor está diseñado en C# para compartir código entre el cliente y el servidor. A base de una búsqueda de librerías previamente encontramos tres librerías para JavaScript (Acorn, Acorn-parser y js-interpreter) y dado que pueden ser separado el sistema, se decidió hacerlo un microservice a lo que se mandarán peticiones una API REST. Adicionalmente, es necesario para algunos puzzles tener un estado del editor del código, se creó una pequeña herramienta donde podemos tener un editor de código Blocky y simplemente obtenemos el código XML del programa del editor para ese tipo de puzzles.

Capítulo 4
Resultados y Discusiones

[En este capítulo se presentan los resultados obtenidos correspondientes al proyecto descrito en el capítulo anterior. Los resultados se pueden presentar en tablas o gráfica y deben ser redactados y organizados de tal manera que sea fácil de comprender por los lectores.

La los resultados no se explican por si mismo, por lo que es necesario una discusión que los explique y muestre cómo ayudan a resolver el problema definido en el capítulo 1. La discusión puede mencionar someramente los resultados antes de discutirlos, pero no debe repetirlos en detalle. No prolongues la discusión citando trabajos "relacionados" o planteando explicaciones poco probables. Ambas acciones distraen al lector y lo alejan de la discusión realmente importante. La discusión puede incluir recomendaciones y sugerencias para investigaciones futuras, tales como métodos alternos que podrían dar mejores resultados, tareas que no se hicieron y que en retrospectiva debieron hacerse, y aspectos que merecen explorarse en las próximas investigaciones.]

Capítulo 5
Conclusiones

[Estos son los enunciados más importantes y más fuertes que se deben realizar acerca de los resultados y discusiones. Las conclusiones deben resumir el contenido y el propósito del proyecto. Se debe hacer énfasis en lo que queremos que se recuerde acerca del proyecto y realizar una síntesis de los resultados que se derivaron de los objetivos específicos trazados inicialmente (y que dieron respuesta a las preguntas de investigación si es que existen). No deben aparecer elementos nuevos o que no fueron discutidos, por ejemplo nuevos resultados observados en otros trabajos. Las conclusiones se refieren única y exclusivamente al proyecto desarrollado.

5.1. Con respecto a las preguntas de investigación

5.2. Con respecto al objetivo de la investigación

5.3. Recomendaciones para futuras investigaciones

[La forma más simple de presentar las conclusiones es enumerándolas consecutivamente, pero podrías optar por recapitular brevemente el contenido del artículo, mencionando some-

ramente su propósito, los métodos principales, los datos más sobresalientes y la contribución más importante de la investigación. La sección de conclusiones no debe repetir innecesariamente el contenido del resumen.]

Bibliografía

[1] “Learning Legendario, "¿Qué es el Aprendizaje justo a tiempo?
”.

[2] Logo Foundation, “Logo History”.

[3] D Hemmendinger, “Pascal”.

[4] J Wolfe, “A brief history of Python”.

[5] D Weintrop, “Block-based Programming in Computer Science Education”.

[6] J Burfoot, “What is LEGO WeDo?”.

[7] L.E.G.O., “LEGO® Education WeDo 2.0 Core Set by LEGO® Education”.

[8] A.Inge Wang, “The Wear Out Effect of a Game-based Student Response System”.

[9] “Mybridge, "12 Free Games to Learn Programming.
”.

[10] I C Mow, “Issues and Difficulties in Teaching Novice Computer Programming”.

[11] N Monjelat, L M Zaballos, and P Lacasa, “Procesos de Resolución de ProblemasyVideojuegos:el Caso de Sim City Creator”.

[12] Gamelearn, “¿Qué es Game-based learning?”.

[13] Gamelearn, “Todo lo que necesitas saber sobre los serious games y el game-based learning, explicado con ejemplos”.

[14] A Levasseur, “The Case for Videogames as Powerful Tools for Learning”.

[15] Cornell University, “Collaborative Learning,”.

[16] L Romano, L Papa, and E., “Saulle, "6 Awesome Cooperative Classroom Games ”.

[17] A.López Najera, “"Kanban ”.

[18] Unity Technologies, “Unity 2019: Performance by default, high-fidelity real-time graphics, and artist tools”.

[19] K Finley, “What Exactly Is Github Anyway?”.

Apéndice A
Nombre del Apéndice

[Sustituye este texto. En esta sección opcional se deberá incluir información secundaria o material importante que es muy extenso. El apéndice se coloca después de la literatura citada. Ejemplos de información que puede colocarse en el apéndice: una lista de universidades visitadas; los datos obtenidos de todas las repeticiones del experimento; derivaciones matemáticas extensas; todos los resultados del análisis estadístico (incluyendo quizás los no significativos) y mapas de distribución para cada fenómeno estudiado; listados completos de código fuente; etc.]